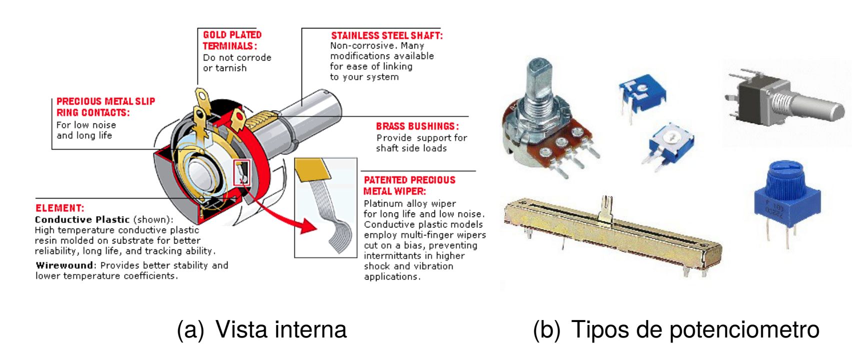
Creative Technologies in the Classroom: Potenciometro y Comunicación Serial

Colegio Seminario Diocesano de Duitama - M&T

1. Potenciometro

Un potenciometro es utilizado para ajustar la sensibilidad de sensores o como resistencia variable en un circuito.

Un potenciometro tiene tres terminales, las terminales derecha e izquierda conectan con los extremos del elemento resistivo interno, el terminal central esta conectado con un contacto conocido como *wiper* que se desplaza de extremo a extremo en contacto con el material resistivo a medida que se rota el eje o se mueve el deslizador variando así el valor de resistencia ([1], pág. 90).



(a) Vista interna (b) Tipos de potenciometro

Figura 1: Vista interna de un potentiometro de eje y diferentes tipos de potenciometros

2. Comunicación Serial

En general, en electrónica existen dos tipos de comunicación, en paralelo y en serie, la comunicación en paralelo puede transmitir 8, 16 o más paquetes de bits simultáneamente haciendo uso del doble número de puertos I/O lo cual es un serio impedimento para su uso en dispositivos con un número de puertos I/O limitado como lo es arduido, la comunicación serial transmite paquetes de un bit de información uno seguido del otro (en "fila") haciendo uso de dos canales I/O, RX (recepción) y TX (transmición).

Históricamente los dos tipos de puertos han convivido en los computadores, dedicando los puertos paralelos a la transmición de grandes paquetes de datos, sin embargo debido al aumento en la velocidad de los procesadores, los puertos serie han desplazado a los puertos paralelos.

Antes de establecer comunicación serial entre cualquier par de sistemas es necesario asegurarse que los puertos son compatibles, esto es, que ambos puertos soporten el mismo voltaje

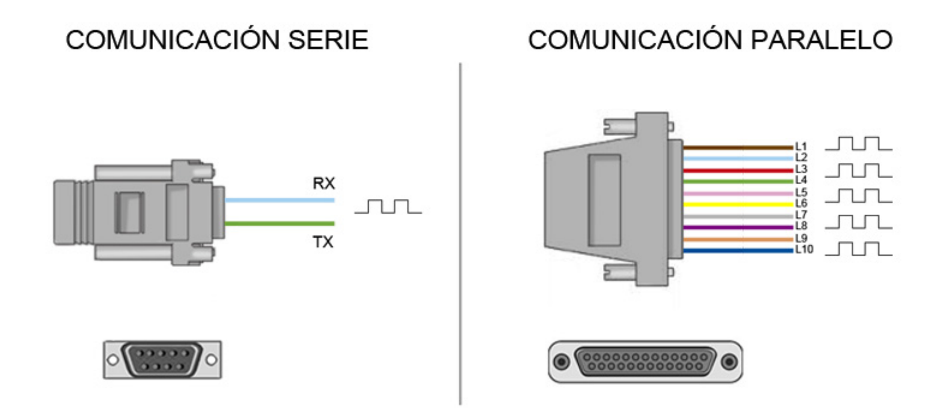


Figura 2: puertos de comunicación

y por ende la misma velocidad de transmición, esta velocidad esta dada en baudios (byts por segundo).

Las placas Arduino Uno y mini cuentan con una unidad UART (Universal Asynchronous Receiver / Transmitter) que opera a nivel TTL 0 / 5V (transistor - transistor logic) lo que lo hace completamente compatible con el puerto USB de los computadores, Arduino Mega y Due cuentan con 4 unidades UART TTL 0 / 5V.

2.1. Monitor Serial

Arduino cuenta con una interfaz de comunicación serial a través de puerto USB llamada `monitor serial`, ésta puede ser accedida desde el botón ubicado en la esquina superior derecha de Arduino IDE.

2.1.1. Arduino Dice Hola Mundo

Un primer ejercicio de comunicación serial entre Arduino y el computador es hacer que arduino salude, para ello es suficiente el Listing 1.

```
1 void setup(){
2   Serial.begin(9600);
3 }
4
5 void loop(){
6   Serial.println("Hola Mundo");
7   delay(1000);
8 }
```

Listing 1: Sketch Hola Mundo

En el Listing 1, la línea 2 establece la velocidad de comunicación a 9600 baudios, es importante observar que en el monitor serial la velocidad de comunicación este definida igual o de lo contrario no podrán entenderse Arduino y el computador.



Figura 3: Monitor Serial

La línea 6 del Listing 1, imprime el mensaje `"Hola Mundo"` en el monitor serial usando un salto de línea cada vez, es decir cada nuevo mensaje se imprime en una nueva línea, aparece un nuevo saludo cada 1000 milisegundos (`delay(1000)`);

2.2. Pines Análogos

Mientras que los pines digitales dan una lectura de voltaje de 0 Volts o 5 Volts, los pines análogos dan valores de voltaje en un rango de 0 - 5 volts a través de un dispositivo interno en Arduino que se llama `conversor analógico - digital`, este dispositivo se encarga de convertir el rango de 0 - 5 volts en un número entero en el rango de 0 (0V) - 1023 (5V), las entradas análogas en Arduino están identificadas con la letra A mientras que las salidas analógas (`PWM`), que se estudiarán posteriormente, están identificadas con el símbolo $\sim$ (circunflejo).

2.2.1. Leyendo un potenciometro

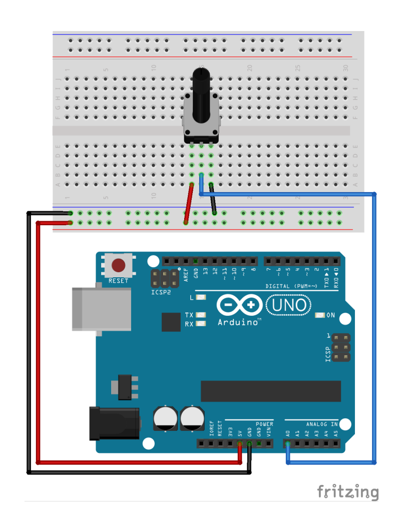


Figura 4: Montaje de un potenciometro usando Arduino Uno

En este sketch simplemente se escribirá en el monitor serial, el valor que lee el pin análogo A0 al cual esta conectado un potenciometro, esta valor no corresponde al valor de resistencia del potenciometro sino a un valor entero entre 0 - 1023; luego de hacer el montaje de la Figura

4 y cargar el siguiente cógio a la placa Arduino puede abrirse el monitor serial y rotar el eje del potenciometro para ver cómo cambia el valor en el pin A0.

```
1  int sensor = A0;
2  int valSensor = 0;
3
4  void setup(){
5    Serial.begin(9600);
6  }
7
8  void loop(){
9    valSensor = analogRead(sensor);
10   Serial.println(valSensor);
11   delay(100);
12 }
```

Listing 2: Sketch para leer un potenciometro

Si lo que se quisiere es leer el valor de la resistencia en el potenciometro a medida que se rota el eje, sería necesario *mapear* el valor del sensor que esta en el rango 0 - 1023 al rango de resistencia del potenciometro, así, por ejemplo, si se esta utilizando un potenciometro de 250 Kohm se sustituirá la línea 9 del sketch anterior por `valSensor = map(analogRead, 0, 1023, 0, 250);` así, en el monitor serial empezarán a aparecer valores enteros en el rango 0 - 250 en lugar de 0 - 1023 como en el sketch original.

3. Hazto tu mismo

3.1. Comunicación Arduino - Processing

`Processing` es un entorno de aprendizaje de codificación (Software Libre) nacido en 2001 pensado para artistas visuales; dentro de sus muchas característica esta el ser altamente compatible con `Arduino`, entre otras cosas, puede establecerse un entorno visual de las aplicaciones de hardware desarrolladas sobre Arduino.

En esta actividad lo que vas a hacer es construir un pequeño juego que presentará en pantalla dos carreteras serpenteantes y en cada una un coche que deberás conducir mediante un par de potenciometros conectados a Arduino para evitar que se salgan de la carretera; luego utilizarás el mismo concepto de comunicación serial esta vez entre Arduino y Processing. Para representar las carreteras aprovecharás algunos conceptos de la representación del lugar geométrico de curvas sinusoidales.

3.1.1. Materiales

Para esta actividad necesitarás

- 1 Placa Arduino
- 1 Breadboard

- 6 Wirejumpers
- 2 Potenciometros
- Deberás también tener instalado Processing en tu computador.

Para el montaje, simplemente deberás agregar un potenciometro más al mostrado en la Figura 4, éste segundo potenciometro conectado al pin análogo 1 (puede utilizar cualquier otro pin análogo siempre que lo tengas presente para modificar el código).

3.1.2. El código del lado de Arduino

En el `Listing 3` aparte de algunas líneas que ya se mencionaron, para establecer comunicación serial se incluye la función `establishContact` que simplemente envía una letra "A" mientras que no haya ningún mensaje presente en el puerto, en el bucle `loop`, si hay algún mensaje presente en el puerto (línea 13), mapea los valores leídos en los pines análogos 0 y 1 a valores entre 0 y 255 y los envía en orden.

```
1 int sensor0 = A0;
2 int valSensor0 = 0;
3 int sensor1 = A1;
4 int valSensor1 = 0;
5 int inByte = 0;
6
7 void setup() {
8   Serial.begin(9600);
9   establishContact();
10 }
11
12 void loop() {
13   if (Serial.available() > 0){
14     inByte = Serial.read();
15     valSensor0 = map(analogRead(sensor0), 0, 1023, 0, 255);
16     delay(10);
17     valSensor1 = map(analogRead(sensor1), 0, 1023, 0, 255);
18     Serial.write(valSensor0);
19     Serial.write(valSensor1);
20     delay(10);
21   }
22 }
23
24 void establishContact(){
25   while (Serial.available() <= 0){
26     Serial.print('A');
27     delay(300);
28   }
29 }
```

Listing 3: Sketch para enviar por puerto USB el valor de 2 potenciometros

El `Listing 3` puede modificarse fácilmente para agregar más datos enviados.

3.1.3. El código del lado de Processing

En Processing el código se divide en dos partes, el Listing 4 presenta la clase `street` que se encarga de dibujar mediante elipses los bordes de las carreteras mostradas en patanlla y el Listing 5 muestra el código del juego propiamente dicho, en el Listing 5 especial atención se presta a las líneas 1 y 2 en las que importa la librerá de comunicación serial y se define un objeto que determina el puerto de comunicación. En las 12 a 14 se definen, la variable `firstContact` para establecer cuando se ha establecido comunicación, un vector `valorSerie` que contendrá los valores leídos desde Arduino y una variable de control `seriesContados` que determina el número de datos recibidos. La línea 18, dentro de `setup` asigna el puerto utilizado por Arduino al objeto `portUSB` y finalmente la función `serialEvent` entre las líneas 54 a 72, lee los datos enviados desde Arduino, los mapea y asigna a las variables correspondientes `absL` y **absR** que determinan la posición horizontal de los coches en pantalla.

```
class street{
  float x, y, A, B, C, D, k;

  street(float X, float Y, float T, float dh, float dv){
    C = dh;
    B = T;
    D = dv;
    x = X;
    y = Y;
  }

  void dibuja(float A){
    noStroke();
    fill(52, 52, 179);
    k = A * sin(B * (x += .003) + C) + D; // Abscisa de la ellipse
    ellipse(k, y, 2, 2);
  }

}
```

Listing 4: Clase `street` de Processing

```
1 import processing.serial.*;
2 Serial portUSB;
3 street[] calle;
4 street[] calle1;
5 street[] calle2;
6 street[] calle3;
7
8 int time;
9 float A, absL, absR;
10
11 // Variables de comunicacion
12 boolean firstContact = false;
13 int[] valorSerie = new int[2];
```

```
int seriesContados = 0;

void setup(){
  size(400, 500);
  portUSB = new Serial(this, "/dev/ttyACM0", 9600);
  calle = new street[height];
  calle1 = new street[height];
  calle2 = new street[height];
  calle3 = new street[height];
  for (int i = 0; i < calle.length; i++){
    calle[i] = new street(i * 0.01, i, 5, 6, 75);
    calle1[i] = new street(i * 0.01, i, 5, 6, 125);
    calle2[i] = new street(i * 0.01, i, 4, -1, 275);
    calle3[i] = new street(i * 0.01, i, 4, -1, 325);
  }
  A = 35;
  absL = 0;
  absR = 0;
}

void draw(){
  background(#A6BCD8);

  for (int i = 0; i < calle.length; i++){
    calle[i].dibuja(A);
    calle1[i].dibuja(A);
    calle2[i].dibuja(A);
    calle3[i].dibuja(A);
  }

  if (absL <= calle[250].k || absL >= calle1[250].k){
    fill(#FA5305);
  }
  else{
    fill(#259004);
  }
  rect(absL, height / 2, 5, 7);
  if (absR <= calle2[250].k || absR >= calle3[250].k){
    fill(#FA5305);
  }
  else{
    fill(#259004);
  }
  rect(absR, height / 2, 5, 7);
}

void serialEvent(Serial portUSB){
  int inByte = portUSB.read();
  if (firstContact == false){
    if (inByte == 'A'){
      portUSB.clear();
```

```
      firstContact = true;
      portUSB.write('A');
    }
  }
  else{
    valorSerie[seriesContados] = inByte;
    seriesContados ++;
    if (seriesContados > 1){
      absL = map(valorSerie[0], 0, 255, 0, 200);
      absR = map(valorSerie[1], 0, 255, 200, 400);
      portUSB.write('A');
      seriesContados = 0;
    }
  }
}
```

Listing 5: Sketch `Carretera` en Processing

Los archivos `street.pde` y `carretera.pde` deben encontrarse en la misma carpeta contenedora para que processing pueda ejecutarlos con éxito.

En los materiales de formación dedicados a Processing puedes encontrar mayor información sobre cómo construir el código correspondiente de los Listing 4 y 5.

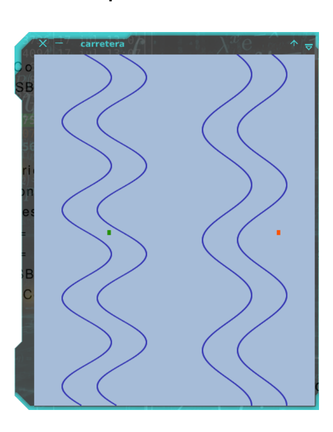


Figura 5: Muestra de la ventada del juego

Referencias

[1] C. Platt, *Encyclopedia of electronic components volume 1: Resistors, capacitors, inductors, switches, encoders, relays, transistors*, Encyclopedia of Electronic Components, O'Reilly Media, Incorporated, 2012.

Lic. Fausto Mauricio Lagos Suárez
Mg. Ingeniería computacional y matemática

